---

1. (UFRN 2017: Engenheiro - Neuroengenheira) Uma sequência de números é um Tipo Abstrato de Dados (TAD) que representa um conjunto finito de valores ordenados, no qual um valor pode ocorrer em duplicidade. Considere as seguintes afirmações sobre a implementação de uma sequência de números utilizando arranjos e listas ligadas:

   I Arranjos permitem acesso a qualquer elemento da sequência com complexidade de tempo média constante.

   II Listas ligadas não permitem a inserção de um elemento no início da sequência com complexidade de tempo média constante.

   III Listas ligadas requerem que a sequência seja armazenada em uma faixa contínua de endereços de memória.

   IV Arranjos não permitem a inserção de um elemento no meio da sequência com complexidade de tempo média constante.

   Estão corretas as afirmações

   (a) I e II.

   (b) II e III.

   (c) I e IV.

   (d) III e IV.

2. (UFVJM-MG 2017: Analista de Tecnologia da Informação) Qual é o tipo de algoritmo de ordenação que tem como princípio percorrer o vetor diversas vezes, a cada passagem fazendo o maior elemento se mover para o final da estrutura?

   (a) *Double sort*

   (b) *Heap sort*

   (c) *Merge sort*

   (d) *Bubble sort*

3. (UFVJM-MG 2017: Analista de Tecnologia da Informação) Qual é o método de ordenação mais eficiente entre os listados a seguir?

   (a) $O(n * n^2)$

   (b) $O(n^2)$

   (c) $O(2^n)$

   (d) $O(n^n)$

4. (TRE-RJ 2017: Técnico Judiciário - Programação de Sistemas) Segundo a análise do trecho de algoritmo a seguir, conclui-se que se trata de um algoritmo de ordenação do tipo:

```
1   declare X[5], i, j, eleito numérico
2   para i ← 1 até 4 faça
3     inicio
4     eleito ← X[ i ]
5     j ← i – 1
6     enquanto (j >= 0 E X[ j ] > eleito)
7       inicio
8           X [ j + 1 ] ← X [ j ]
9           j ← j – 1
10      fim
11      X[j + 1] ← eleito
12    fim-enquanto
13  fim-para
```

(a) *Quick sort*

(b) *Bubble sort*

(c) *Insertion sort*

(d) *Selection sort*

5. (TRE-RJ 2017: Técnico Judiciário - Programação de Sistemas) O trecho de algoritmo a seguir corresponde ao método de ordenação do tipo:

```
1   declare X[5], n, i, aux numérico
2   para n ← 1 até 5 faça
3      início
4          para i ← 0 até 3 faça
5              início
6                  se (X [i] > X[i+1] )
7                      então início
8                          aux X[i]
9                          X [i] ← X[i + 1]
10                         X[i+1] ← aux
11             fim
12     fim
13  fim
```

(a) *Quick sort*

(b) *Merge sort*

(c) *Bubble sort*

(d) *Insertion sort*

6. (IFPE 2016: Professor - Informação e Comunicação) Os algoritmos de busca e ordenação são bem conhecidos no contexto da computação. Embora muita coisa tenha evoluído na área, alguns algoritmos são clássicos. Nesse sentido, de acordo com o método *algoritmoX* abaixo, é possível afirmar que ele é um algoritmo de:

```
public class Q3 {
	int algoritmoX (int chave, int vetor[], int limInferior, int
limSuperior) {
		int indice = (limInferior + limSuperior)/2;
		if (vetor[indice] == chave)
			return indice;
		if (limInferior >= limSuperior)
			return -1;
		else
			if (vetor[indice] < chave)
				return algoritmoX(chave, vetor, indice+1,
limSuperior);
			else
				return algoritmoX(chave, vetor,
limInferior, indice-1);
	}

	public static void main(String[] args){
		//...
	}
}
```

(a) Ordenação *bubblesort*

(b) Busca sequencial ou linear

(c) Busca binária

(d) Ordenação *quicksort*

7. (CODEBA 2016: Analista Portuário - Analista de Tecnologia da Informação) Considere um *array* R que contém 1.000.000 de chaves ordenadas. Assinale o número máximo de acessos a R necessários para encontrar uma determinada chave.

(a) 10

(b) 20

(c) 40

(d) 80

(e) 160

8. (IF-PE 2016: Professor - Informação e Comunicação) Uma das atividades mais executadas em programas de computador é a ordenação. Na literatura, existem várias abordagens para se implementar algoritmos de ordenação. Assinale a alternativa que apresenta a estratégia de ordenação implementada no algoritmo abaixo:

(a) *Bubble Sort*
(b) *Quick Sort*
(c) *Bucket Sort*
(d) *Radix Sort*
(e) *Merge Sort*

```
public class Ordenacao {
    private int[] numbers;
    private int[] helper;

    private int number;

    public void ordenar(int[] values) {
        this.numbers = values;
        number = values.length;
        this.helper = new int[number];
        metodo1(0, number - 1);
    }

    private void metodo1(int low, int high) {
        if (low < high) {
            int middle = low + (high - low) / 2;
            metodo1(low, middle);
            metodo1(middle + 1, high);
            metodo2(low, middle, high);

        }
    }

    private void metodo2(int low, int middle, int high) {

        for (int i = low; i <= high; i++) {
            helper[i] = numbers[i];
        }

        int i = low;
        int j = middle + 1;
        int k = low;
        while (i <= middle && j <= high) {
            if (helper[i] <= helper[j]) {
                numbers[k] = helper[i];
                i++;
            } else {
                numbers[k] = helper[j];
                j++;
            }
            k++;
        }
        while (i <= middle) {
            numbers[k] = helper[i];
            k++;
            i++;
        }
    }
}
```

9. (COPESE-UFPI 2017: Analista de Tecnologia da Informação) Um conjunto ordenado de itens a partir do qual podem ser eliminados itens em uma extremidade e no qual podem ser inseridos itens na outra extremidade é denominado de:
   (a) fila.
   (b) pilha.
   (c) lista simples.
   (d) lista encadeada.
   (e) árvore.

10. (IESES 2017: Analista de Gestão - Analista de Sistemas) Considerando as definições para listas (pilhas e filas), assinale a alternativa correta.
   (a) Uma lista é um tipo de fila que se caracteriza por considerar que o primeiro elemento a entrar é o primeiro a sair.
   (b) Lista é um conjunto de filas e pilhas e se compõe por elementos que podem ser ligados ou não.
   (c) Uma lista pode ter uma configuração que possa ser uma arvore balanceada ou não.
   (d) Lista é uma sequência finita de elementos ligados entre si. Podem ser organizada de tal forma que implemente uma fila ou uma pilha.

11. (TCE-SE 2015: Analista de Tecnologia da Informação-Desenvolvimento) A tabela a seguir deve ilustrar uma lista duplamente encadeada de cores, estruturada sobre os cinco elementos de um vetor.

Elemento	Anterior	Seguinte	Cor
1	4	2	...
2	1	3	...
3	2		...
4	5	1	...
5		4	...

   Dado que a ordem correta das cores é Marrom-Verde-Azul-Vermelho-Amarelo, a coluna Cor, na tabela acima, deveria apresentar, de cima para baixo, os seguintes valores:
   (a) Marrom-Vermelho-Amarelo-Azul-Verde;
   (b) Marrom-Vermelho-Amarelo-Azul-Verde;
   (c) Amarelo-Azul-Marrom-Vermelho-Verde;
   (d) Azul-Vermelho-Amarelo-Verde-Marrom;
   (e) Verde-Azul-Vermelho-Marrom-Amarelo.

12. (IF Farroupilha - RS 2016: Docente - Informática Geral) Uma lista encadeada é o melhor e mais simples exemplo de estrutura de dados dinâmica que utiliza ponteiros em sua implementação. O primeiro nó de uma lista é conhecido como cabeça, do inglês *head*. Se uma lista está vazia, então o valor da cabeça ou *head* é nulo, do inglês *NULL*. Com base nisso, analise os trechos de código abaixo que implementam algumas ações sobre uma lista encadeada em linguagem C.

```
typedef struct node
{
 int data;
 struct node *next;
} node_t;
```

```
void fun1(node_t * head, int val) {
  node_t * current = head;
  while (current->next != NULL) {
     current = current->next;
  }
  current->next =  malloc(sizeof(node_t));
  current->next->val = val;
  current->next->next = NULL;
}
```

```
node_t * fun2(node_t * head, int value) {
   node_t * current = head;

   while (current != NULL) {
      if (current->data == value)
         return current;
      current = current->next;
   }
   return NULL;
}
```

```
int fun3(node_t ** head) {
 int retval = -1;
   node_t * next_node = NULL;

   if (*head == NULL) {
      return -1;
   }
   next_node = (*head)->next;
   retval = (*head)->val;
   free(*head);
   *head = next_node;
   return retval;
}
```

```
void fun4(node_t ** head, int val) {
   node_t * new_node;
   new_node = malloc(sizeof(node_t));

   new_node->val = val;
   new_node->next = *head;
   *head = new_node;
}
```

Sobre os trechos de código acima, é INCORRETO afirmar que:

(a) A função *fun3* remove um elemento do final da lista encadeada.

(b) A função *fun1* adiciona um novo elemento no final da lista encadeada.

(c) A função *fun4* adiciona um novo elemento no início da lista encadeada.

(d) As funções *fun1* e *fun4* são funções que adicionam novos elementos na lista encadeada.

(e) A função *fun2* pode ser utilizada para encontrar um nó da lista que tenha o valor *value*, que é passado como argumento para a função.

13. (INMETRO 2015: Assistente Executivo em Metrologia e Qualidade - Informática) A descrição de uma determinada estrutura de dados deverá ser implementada. Na descrição apresentada, cada item dessa estrutura contém a informação necessária para alcançar o próximo item. Esse tipo de implementação permite utilizar posições não contíguas de memória, sendo possível inserir e retirar elementos, sem haver a necessidade de deslocar itens seguintes dessa estrutura. Trata-se da estrutura:

    (a) Filas por meio de arranjos.

    (b) Listas por meio de arranjos.

    (c) Filas com estruturas autorreferenciadas.

    (d) Pilhas com estruturas autorreferenciadas.

    (e) Listas com estruturas autorreferenciadas.

14. (SP-URBANISMO 2014: Analista Administrativo) Tem-se uma estrutura de dados do tipo lista encadeada com 10 elementos, em que o primeiro e o último elemento estão ligados entre si. Trata-se de uma estrutura de dados denominada Lista

    (a) Binária

    (b) Balanceada.

    (c) Invertida.

    (d) Encadeada Circular

    (e) Duplamente Encadeada

15. (UNIRIO 2014: Analista Tecnologia da Informação - Desenvolvimento de Sistemas) Sobre listas lineares, é CORRETO afirmar que

    (a) na representação encadeada, um elemento pode ser inserido em qualquer posição da lista sem movimentar os elementos subsequentes de suas atuais posições na memória.

    (b) se os elementos são incluídos em uma lista por uma das extremidades e retirados pela outra, essa lista é uma pilha.

    (c) na representação encadeada, a exclusão de um elemento provoca a movimentação dos demais elementos de suas atuais posições de memória.

    (d) na representação vetorial, a inserção de um elemento em qualquer posição da lista é feita com esforço computacional constante.

    (e) filas podem ser implementadas apenas através da representação vetorial.

16. (Banco da Amazônia 2014: Técnico Científico - Analise de Sistemas) Uma lista duplamente encadeada tem como característica ser formada por elementos que:

    (a) se concatenam de forma circular, de tal maneira que, ao chegar ao final da lista, o próximo elemento volta a ser o primeiro.

    (b) contêm, além de um ou mais campos chave, mais um campo de ponteiro: o próximo, que permite o acesso ao elemento que sucede o atual (o próximo) presente na mesma lista.

    (c) contêm, além de um campo chave, mais um campo de ponteiro: o próximo, que permite o acesso ao elemento que sucede o atual (o próximo) presente na mesma lista, de tal forma que os campos chave estão ordenados, ou seja, a chave do próximo é sempre maior ou igual à chave do atual elemento.

    (d) contêm, além de um ou mais campos chave, dois outros campos de ponteiros: próximo e anterior, que permitem o acesso aos elementos adjacentes (próximo e anterior) presentes na mesma lista.

    (e) estão em posições adjacentes da memória, permitindo o acesso sequencial ao próximo e ao anterior de cada elemento pelo simples uso de um índice.